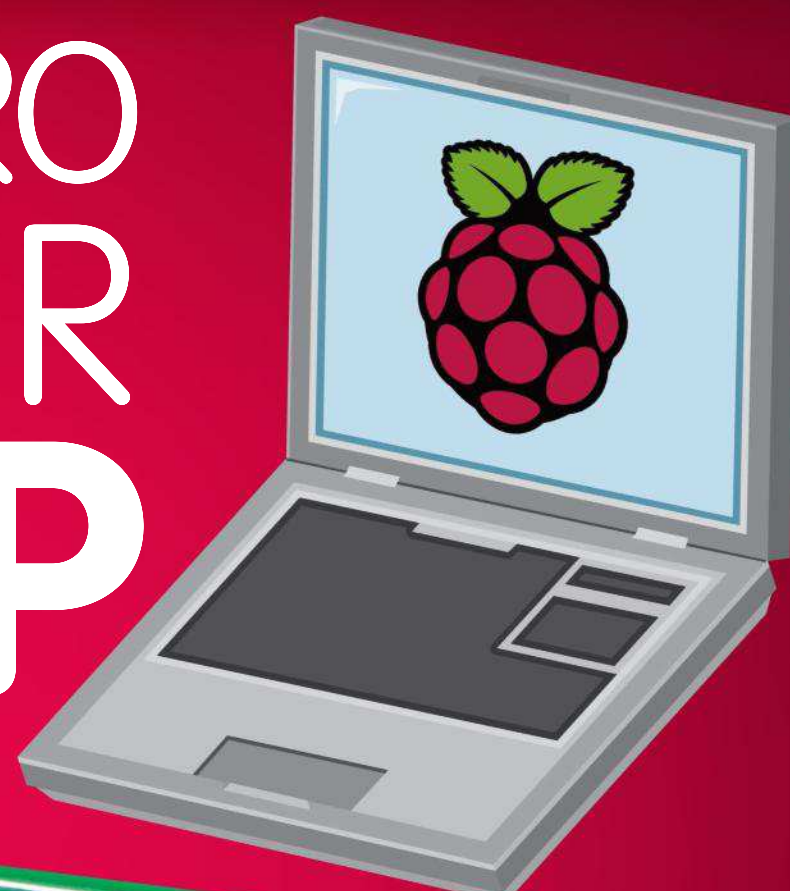
RasPi

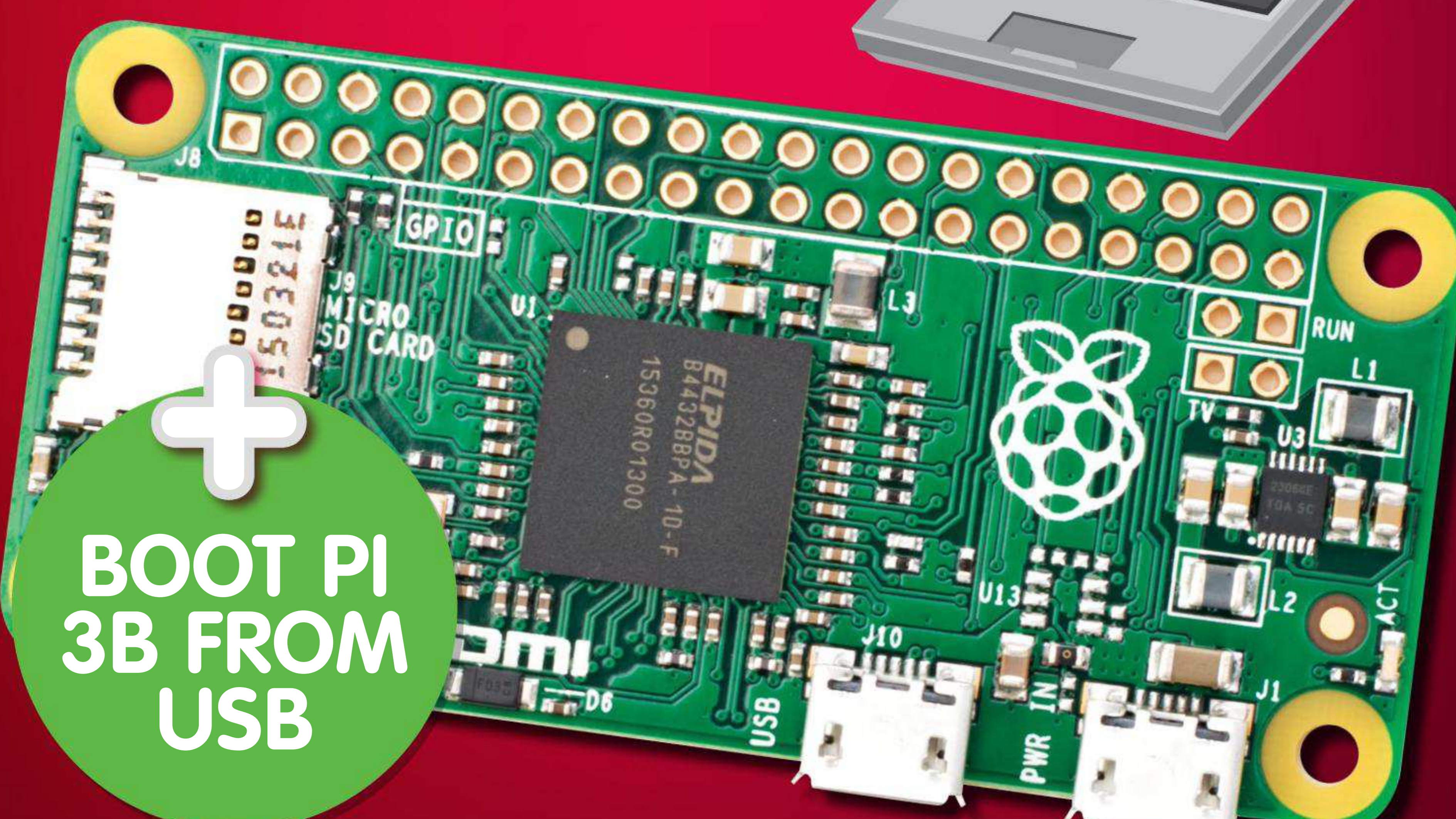
DESIGN
BUILD
CODE

47

Get hands-on with your Raspberry Pi

ACCESS PI ZERO FROM YOUR LAPTOP





+ BOOT PI 3B FROM USB

Plus Retro games on your Pi part 2

Welcome

Raspberry Pi magazine is back again, bringing with it the final part of the tutorial we started last issue that showed you how to turn an old Xbox controller into a retro arcade machine. When you last saw us we'd just finished showing you how to hack the hardware and this issue we're going to show you how to add a bunch of retro ROMS so you can play to your heart's content. Swipe left to get started.

There's plenty of other tutorials that you might find useful too including how to boot a Pi3 B+ from USB and how to access a Pi Zero from your laptop. Finally, our regular Python column will show you how to Stream to Twitch from your Raspberry Pi.

Editor

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

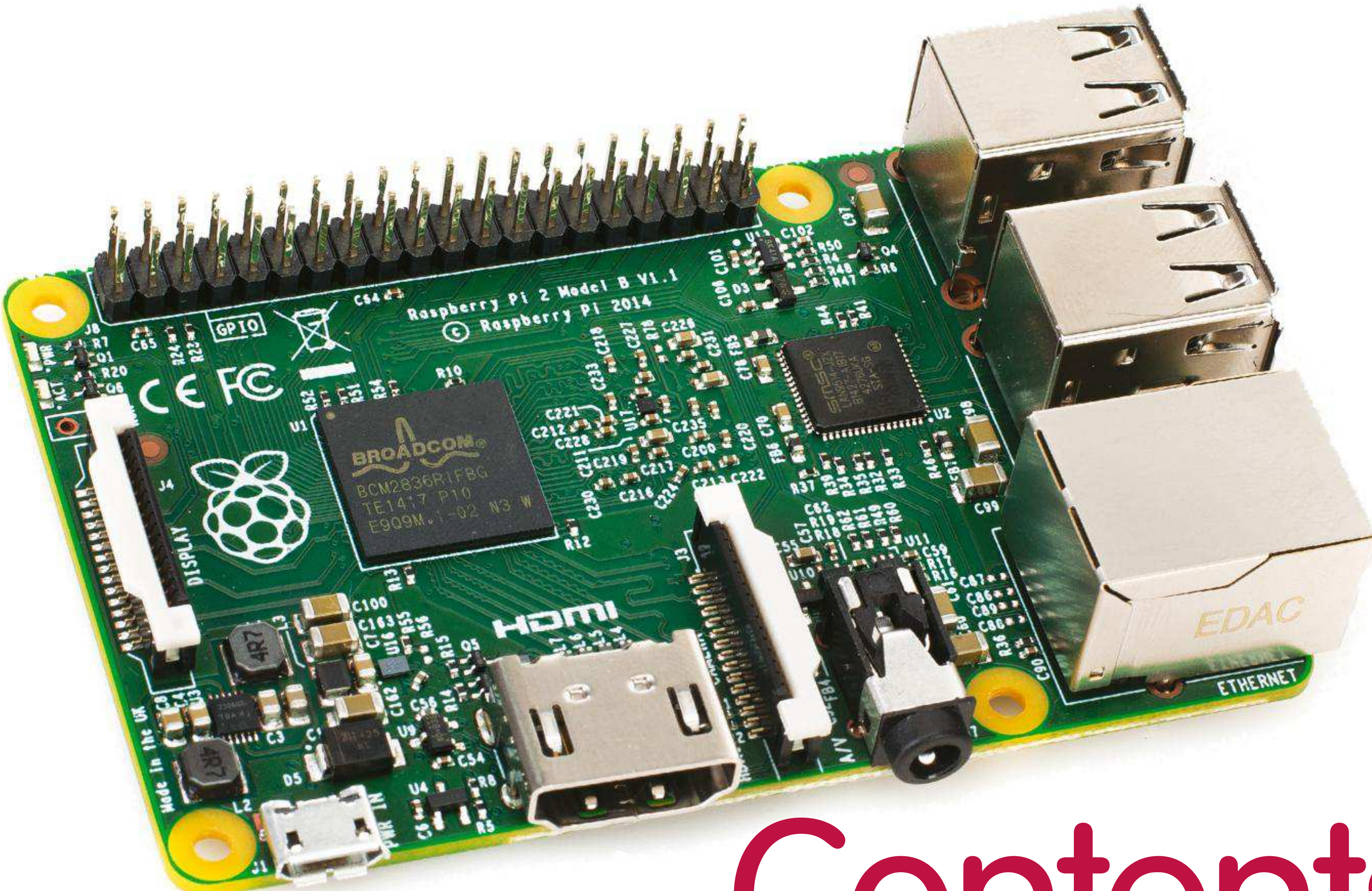
Learn to make and code gadgets with Raspberry Pi

From the makers of

LinuxUser & Developer

Join the conversation at…

@linuxusermag | Linux User & Developer | linuxuser@futurenet.com


Raspberry Pi 2 Model B V1.1
© Raspberry Pi 2014
GPIO
BROADCOM
DISPLAY
HDMI
EDAC
ETHERNET

Contents

Xbox Zero arcade Pt 2

Install an emulator and get retro ROMs up and running

Right, so you've managed to get your Pi safely ensconced in a controller and all wired up – all you need now are some videogames to play.

For this section of the tutorial we're going to be using the RetroPie emulator. By the end of this tutorial, you'll be able to play a number of games directly from your Raspberry Pi, provided that you legally own the ROM files, of course.

The whole process is as easy as installing the software onto your SD card and then copying across any games that you want to play. If you've already got Raspian installed on your Pi, you can install RetroPie alongside it – or you can dedicate the whole disk to the software if you'd rather.

01 Install RetroPie inside Raspbian

If you've already started using your Pi and want to add RetroPie to it, you'll need to install the software from GitHub. The latest instructions can be found at **github.com/RetroPie/RetroPie-Setup.**

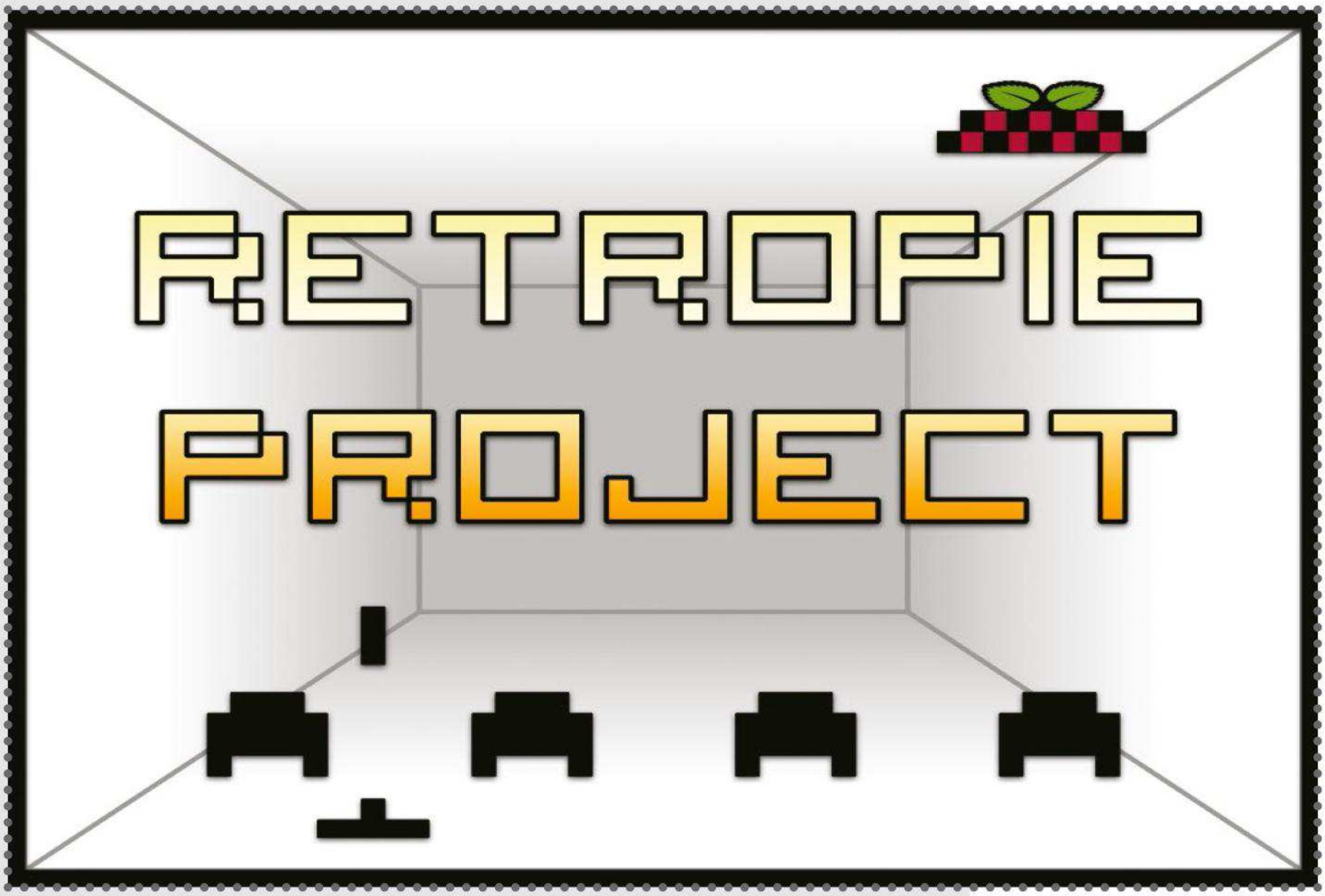
Open up a terminal on your Pi (for example, by SSHing into it from another machine, or by logging in directly to the Pi). Update your repositories and make sure the latest version of the Git software is installed:

```
    sudo apt-get update
    sudo apt-get upgrade
    sudo apt-get install git
```

Download the latest version of the RetroPie setup script:

```
    git clone --depth=1 https://github.com/
RetroPie/RetroPie-Setup.git
```

If you're security-conscious, it's a good idea to check what



Above If you see a splash screen like this when you power on again, the installation worked!

the script does before running it. Once you're ready, you can install it by changing into the correct directory and executing the script:

```
cd RetroPie-Setup
sudo ./retropie_setup.sh
```

The script will take several minutes to run, depending on the speed of your internet connection. It may also ask you for permission to install extra software that is needed – you should allow this. Once fully installed, you will need to reboot your Pi:

```
sudo reboot
```

RetroPie can now be run by typing emulationstation. We'll come on to configuring your setup in just a moment.

02 Install RetroPie onto a blank SD card

If you want your Raspberry Pi Zero to be used solely as a RetroPie machine, this is the choice for you. Be warned: it will completely wipe a micro SD card, so if you're using one you've used before, make sure you back up any important data before starting.

Download the latest version of the software from **http://blog.petrockblock.com/retropie/retropie-downloads**. Make sure you download the correct SD card image for your machine – the image for the Raspberry Pi 2 is not compatible with the Raspberry Pi Zero. Download the Standard version (not the BerryBoot version). The download is an 800MB .gz file. Unzip it and extract the .img file, which will be around 2.6GB.

You'll now need to write this image file onto your micro SD card. This is done in the same way that you would install a normal Raspberry Pi image onto a card. There are slightly different instructions for Linux, Mac and Windows.

03 Linux

Use the Disk Manager to select the image file and the micro SD card. Follow the on-screen instructions until the image has been fully written to the card.

04 Mac

Download the ApplePi Baker from **www.**

What's an emulator?

An emulator is software which lets your computer pretend to be a different sort of computer. It will allow a Raspberry Pi Zero to run software originally designed for the Sega Mega Drive, or Nintendo N64, old DOS-based PCs, etc. Emulators aren't without their problems, though – it's nearly impossible to perfectly recreate a games console in software. Keep in mind that older games may have bugs ranging from minor sound and graphical glitches to full-blown crashes.

tweaking4all.com/hardware/raspberry-pi/macosx-apple-pi-baker. Once you have it installed, you can select the image file and the micro SD card. Follow the on-screen instructions.

05 Windows

Download the Win32 DiskImager from **http://sourceforge.net/projects/win32diskimager**. Once installed, select the image file and the micro SD card. Follow the instructions until the image has been written to the card.

06 Configuring

Right – you're almost ready to play. Put the micro SD card into the Raspberry Pi Zero, hook up the controller USB cable and the HDMI cable. Finally, plug the Pi into the power. It should boot up automatically and, after a few seconds, you'll be greeted with a configuration screen. RetroPie should automatically detect any connected USB game pads and step you through setting up the buttons. Once you've finished, you'll be presented with a screen showing all the choices you made.

07 Set up the disk

Before we get to playing any games, we need to make sure that RetroPie is able to use all the space on the micro SD card. This will allow you to store ROMs and save your games. Select 'RetroPie' from the menu. You'll be presented with several configuration options. Select "Raspberry Pi Configuration Tool RASPI-CONFIG"
You can change the default username and password at a later date; for now just use the controller to select 'Expand Filesystem'. Next, highlight the 'Select' button and click on it. After a short delay, you will see a success screen – press OK and you'll be taken to the configuration screen. Press

right until 'Finish' is highlighted, then click on it. You should now reboot your Raspberry Pi.

08 Adding ROMs

The final step is adding new ROMs. Once you've legally purchased and downloaded ROMs from the internet, you'll need to copy them onto the micro SD card. ROMs are stored in a separate folder for each system. So, for example, you need to place your Sega Master System ROMs in ~/RetroPie/roms/mastersystem/. Once you've installed ROMs, you're ready to play.

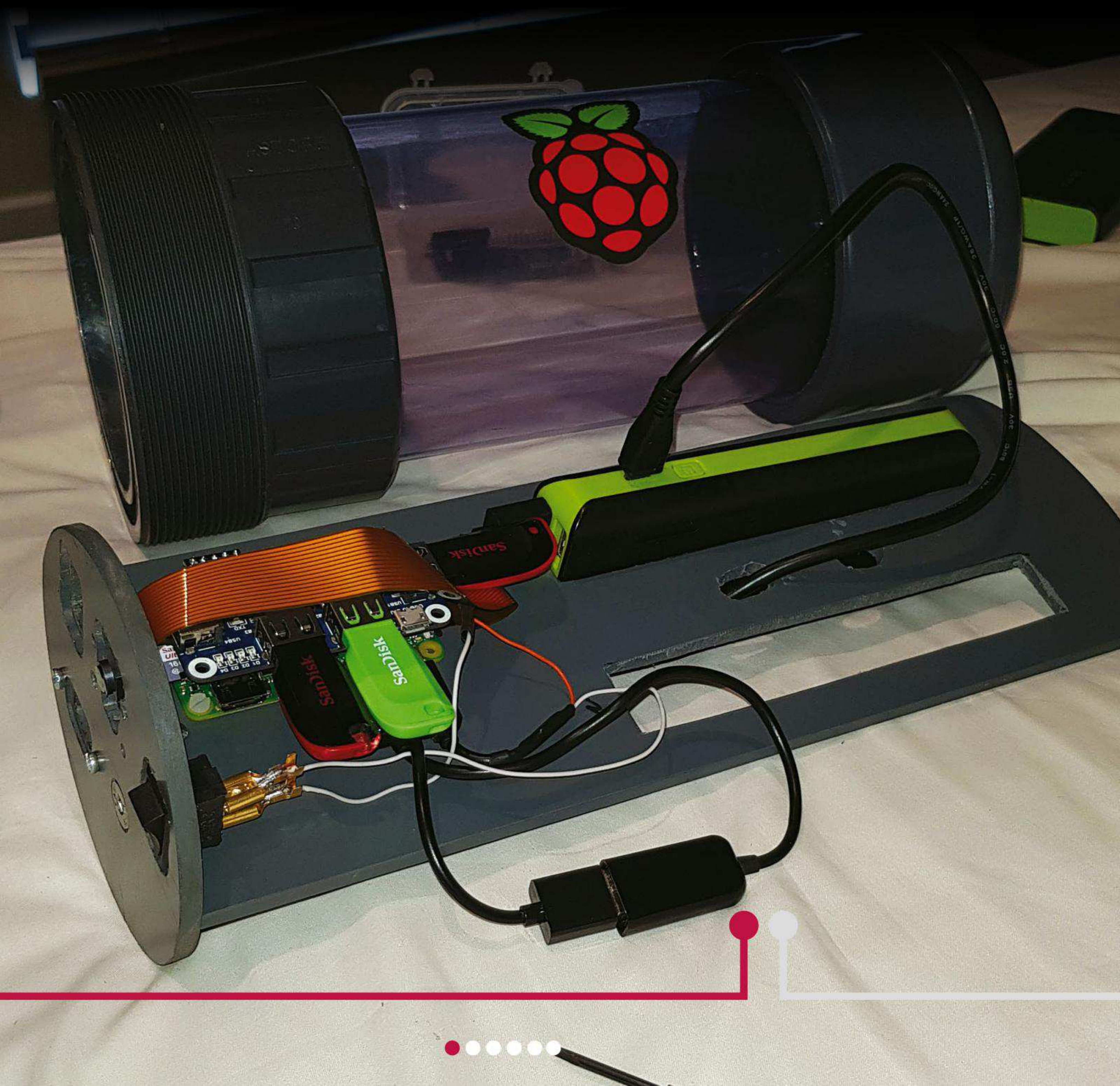
09 Hack your television

Once booted, you'll see a menu with all the available games systems on it. Some emulators will only show up once game ROMs for that system are installed. Scroll until you find the game you want to play – then let rip! You can always return back to RetroPie if you want to change any of the configuration options, or update the software. And that's all there is to it! Time to sit back and play some games. If you want to find out more about the etroPie software, visit http://blog.petrockblock.com/retropie.

Left Energenie's Pi-Mote controller board costs £10, and you can get RC plug sockets with it for an extra £10

PipeCam

Using a Pi to keep an eye on the bottom of the ocean
is simpler than you might think – apart from the leaks



Sometime in 2014, Fred Fourie saw a long-term time-lapse video of corals fighting with each other for space. That piqued his interest in the study of bio-fouling, which is the accumulation of plants, algae and micro-organisms such as barnacles. Underwater documentaries such as Chasing Coral and Blue Planet II further drove his curiosity, and, inspired by the OpenROV project, Fred decided to build an affordable camera rig using inexpensive and easily sourceable components. This he later dubbed PipeCam; head to the project's page (https://hackaday.io/project/21222-pipecam-low-cost-underwater-camera) to read detailed build logs and view the results of underwater tests.

Are power and storage two of the most crucial elements for remote builds such as the PipeCam?
It has been a bit of an ongoing challenge. Initially, I wanted to solve my power issues by making the PipeCam a tethered system, but difficulties in getting a cable into the watertight hull made me turn to a self-contained, battery-powered unit. In the first iterations, I had a small rechargeable lead acid battery and a Raspberry Pi 3, but the current version sports a Pi Zero with a Li-ion power bank. This gives me more than five times the power capacity for a reasonable price. With regards to storage space, I've opted for a small bare-bones USB hub to extend the space with flash drives. There are a few nice Raspberry Pi Zero HATs for this.

What was the most challenging part of the project?
Definitely the underwater housing: I had many leaks.

Fred Fourie
Fred is an electronics technician for an engineering firm in Cape Town, South Africa, that specialises in marine sciences.

The electronics are all off-the-shelf and the online communities has made finding references for the software that I wrote a breeze, but without a good underwater housing the project is… well, literally dead in the water. As of the start of the year I got a friend onboard, Dylan Thomson, to help me with the mechanical parts of the project. Dylan has a workshop with equipment to pressure-test housings (and my calculations). This freed me up to work on the software and electronics.

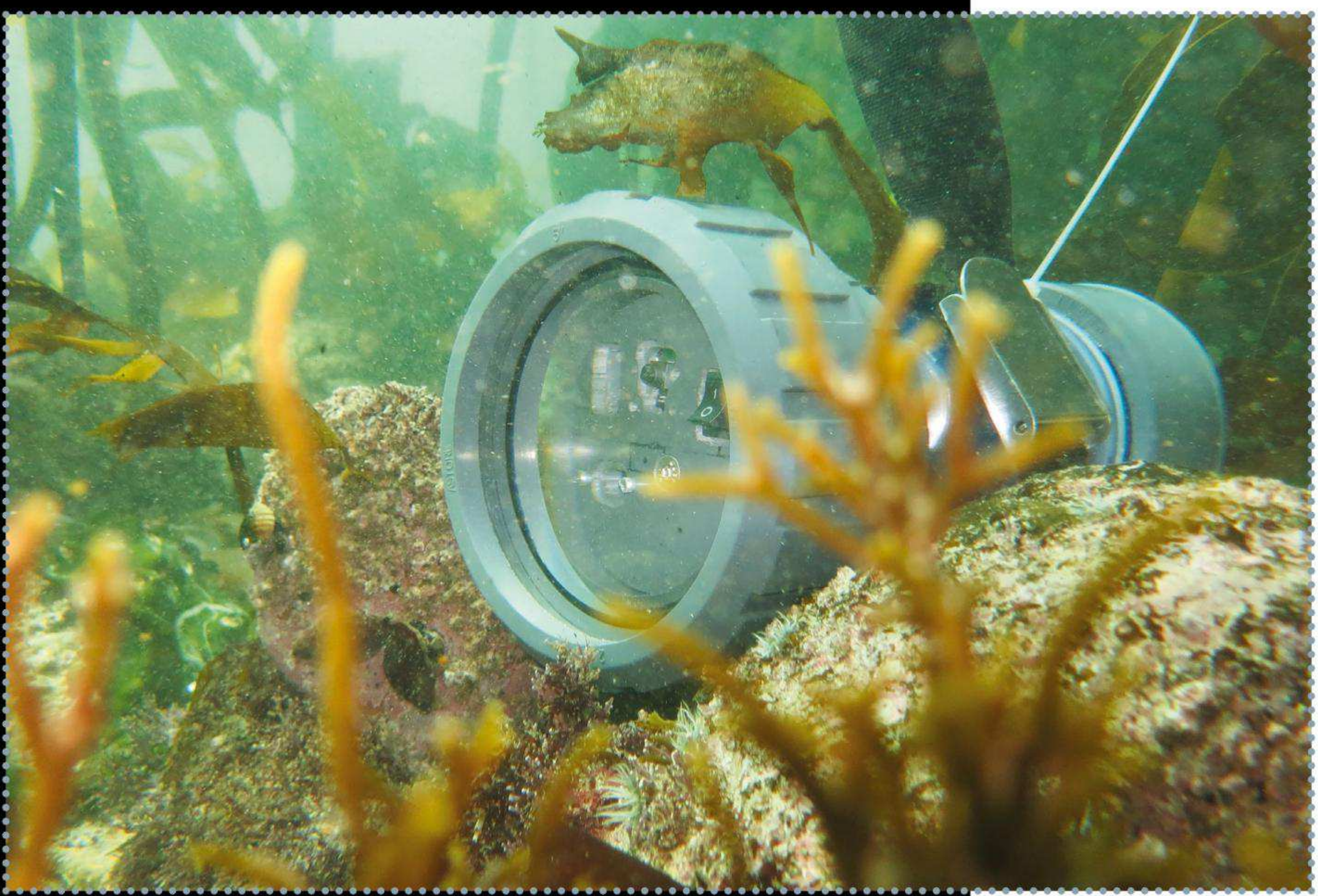
Talking of software, what is the PipeCam running?

I use Raspbian Lite as my base OS. I load up apache2 by default on most projects so I can host 'quick look' diagnostic pages as I tinker. On the PipeCam I installed i2c-tools to set up my hardware clock to keep track of time on longer deployments. I set up my USB drives to be auto-mounted to a specific location. For this I use the blkid to the drive information, and then

add them to the system by editing the /etc/fstab with the drive details and desired location. The main script is written in Python, as it's my home language. The script checks which drive has enough space to record, and depending on the selected mode (video or photo) it then starts the recording or photo-taking process. The script outputs some basic info which I log from the cron call, which is where I set up my recording interval. It's not complicated stuff.

Any particular reason for using the Raspberry Pi?
I know my way around a Linux system far better than I know microcontrollers. The familiarity of the Pi environment made quick setup and experimentation possible. Also, the community support is excellent.

Below Now that he has finalised the design of the build, the next goal is to test the build as much as possible and tweak and refine settings. Fred also intends to give a PipeCam to a few scientists to test in their own fields (or rather water) later this year.



How do you plan to extend the project?

So far the results have been pretty promising. Ultimately the next iteration will aim to increase user-friendliness and endurance. To achieve this there are three sets of modifications I aim to add:

- Make use of the Pi's GPIO to add settings buttons
- Host a user interface webpage on the Pi itself, for system health checks
- Integrate some battery monitoring with use of current- and voltage-sensing circuits, with an light dependent resistor (LDR) to determine if there's enough light to take a picture.

Like it?

Fred has done construction projects in the Antarctic and has worked on space weather on remote islands. He gets excited about biological sciences and large datasets. Follow his adventures on Twittter at **@FredFourie.**

Left While he started the project alone, Fred asked his friend Dylan to join in earlier this year. Dylan handles the mechanical aspect of the project and is in charge of building the robust leak-proof housing, while Fred focuses on software and electronics.

Could you explain the Fritzing schematic you've shared on the project page?

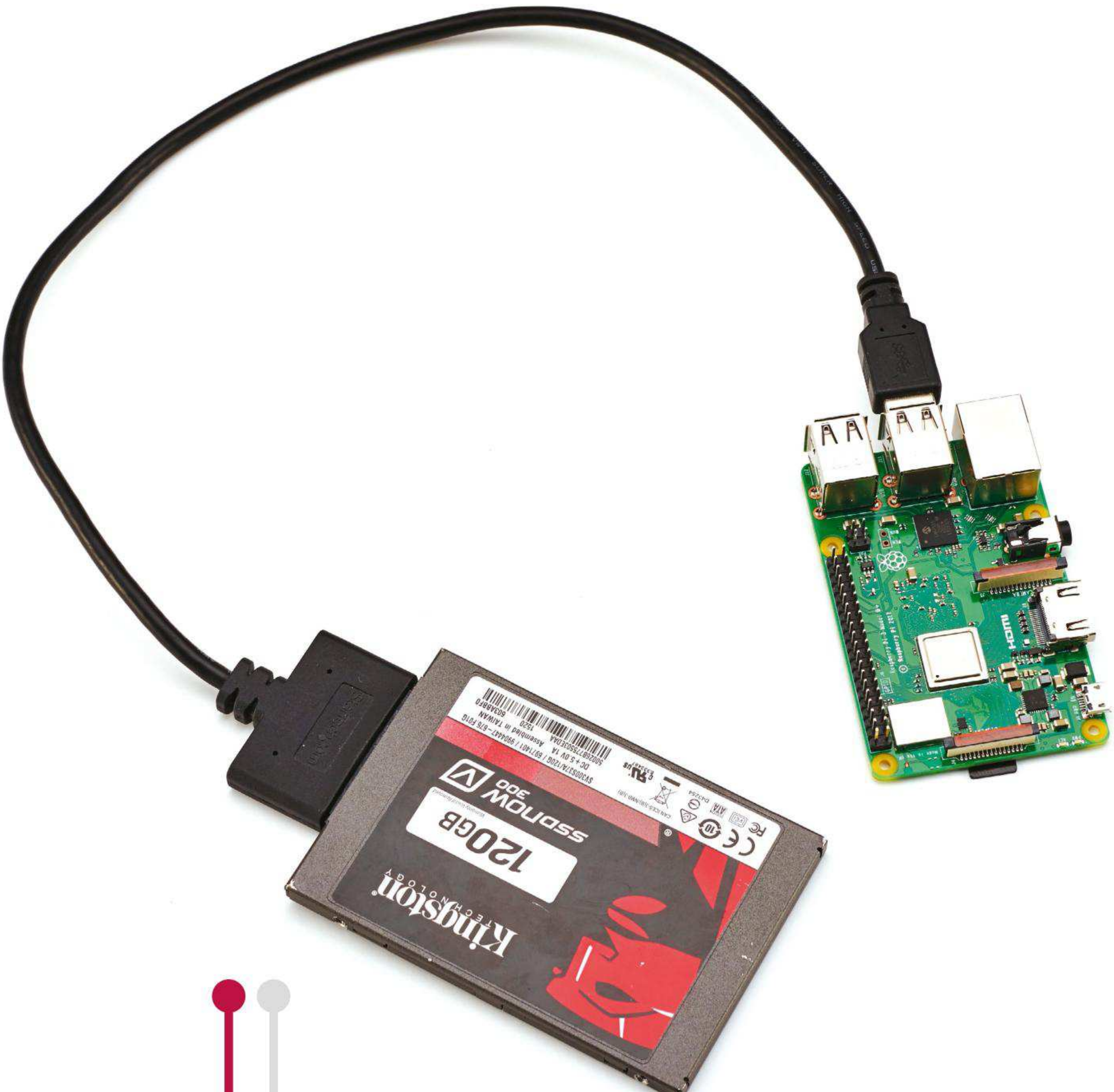
The next iteration is all about reducing the power used in idle times. In the circuit you can see that the main power to the Raspberry Pi is controlled via a relay from a Arduino Nano. The Nano takes inputs from a current sensor, voltage sensor and LDR, and decides from these inputs whether the Pi should be switched on. In addition to the RTC on the Pi, you'll also see a BME280 breakout board to monitor pressure, temperature and humidity, to detect changes associated with leaks. There's also a slide switch to select video or photo mode.

Further reading

Fred is interested in areas where the natural world and electronics meet. He's also been tinkering with machine learning and object detection and suggests there might be some crossover in the future with using object detection. Follow his projects at https://hackaday.io/FredWFourie.

Boot your Pi 3 B+ from USB

Configure and boot up your Raspberry Pi 3 B+ using a USB flash or hard drive



This tutorial explains how to take a USB mass-storage device, such as a flash drive or hard drive and boot up your Raspberry Pi 3 B+ using it. Once everything's configured, there's no longer any need to use an SD card – it can be removed and used in another Raspberry Pi. The benefits of this are that you can increase the overall storage size of the Pi from a standard 4GB-8GB to upwards of 500GB. A further benefit is that the robustness and reliability of a USB storage device is far greater than an SD card, so this increases the longevity of your data.

Before you begin, please note that this setup is still experimental and is developing all the time. Bear in mind too that it doesn't work with all USB mass-storage devices; you can learn more about why and view compatible devices at www.raspberrypi.org/blog/pi-3-booting-part-i-usb-mass-storage-boot.

THE PROJECT ESSENTIALS

Raspberry Pi 3 B+
microSD card
USB storage device

01 How it works

This setup involves booting the Raspberry Pi from the SD card and then altering the config.txt file in order to set the option to enable USB boot mode. This in turn changes a setting in the One Time Programmable (OTP) memory in the Raspberry Pi's system-on-a-chip, and enables booting from a USB device. Once set you can remove the SD card for good. Please note that that any changes you make to the OTP are permanent, so ensure that you use a suitable Raspberry Pi – for example, one that you know will always be able to be hooked up to the USB drive rather than one you might take on the road.

02 Download the latest OS image

You'll obviously need to install the latest version of the OS to make use of this feature, so first open your web browser and head to www.raspberrypi.org/downloads. Select the current Raspbian option and download the 'Stretch with Desktop' image. You can click the link for Release Notes to see all the updates and changes made to the OS with that version. Remember that the file is a zipped file, so you need to extract the IMG from the folder. Open it and drag the file onto your desktop or another folder.

03 Write the OS to the SD card

Now, write the .img image to the SD card. An easy method to do this is with Etcher, which can be downloaded from https://etcher.io. Insert your SD card into your computer and wait for it to load. Open Etcher and click the first 'image' button, select the location of the .img file, then click the 'select drive' button and select the drive letter which corresponds to the SD card. Finally, click the 'Flash!' button to write the image to the card.

04 Write the OS to the USB device

We now need to write the same Raspbian OS image to your USB storage device. You can use the same .img image that you downloaded in step two. Ensure that you have ejected the SD card and load Etcher. Attach the USB storage and once loaded, select the relevant drive from the Etcher menu. Drag the .img image file across as you did in step four. While that's writing, place the SD card into you Raspberry Pi and boot it up ready for the next step.

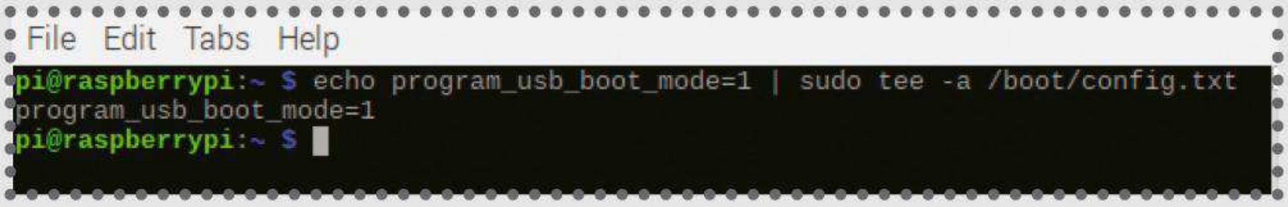
05 What's new

With the release of the new Raspberry Pi 3 B+ the operating system was also updated. This features an upgraded version of Thonny, the Python editor, as well as PepperFlash player and Pygame Zero version 1.2. There's also extended support for larger screens. To use that, from the main menu select the Raspberry Pi configuration option. Navigate to the System tab and locate the 'Pixel Doubling' option. This option draws every pixel on the desktop as a 2x2 block of pixels, which makes everything twice the size. This setting works well with larger screens and HiDPI displays.

06 Configure the Wi-Fi

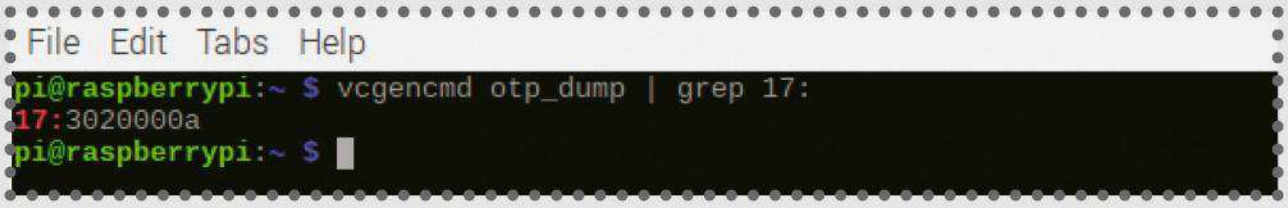
With the latest OS update, Wi-Fi is disabled until the

‘regulatory domain’ is set. This basically means that you have to specify your location (in terms of country) before your Wi-Fi becomes available. Open the main menu and scroll to Raspberry Pi Configuration settings, then select the ‘Localisation’ tab and then ‘Set WiFi Country’. Scroll down the list and select your relevant country for your current location.



07 Configure the USB boot mode

In order to boot your Raspberry Pi from the USB device, you need to alter the config.txt file to stipulate that future boots happen from the USB. Open the Terminal window and type echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt. This adds the line program_usb_boot_mode=1 to the end of /boot/config.txt. This sets the OTP (One Time Programmable) memory in the Raspberry Pi’s SoC to enable booting from the USB device. Remember that this change you make to the OTP is permanent and can’t be undone.



08 Check the configuration

Once you’ve edited the config file, type sudo reboot to reboot your Raspberry Pi. The next step is to check that the OTP has been programmed correctly. Open the Terminal window and enter the following:

vcgencmd otp_dump | grep 17 then press Enter. If the OTP has been programmed successfully, 17:3020000a will be displayed in the Terminal. If it's any different, return to step 7 and re-enter the line of code.

09 Boot from the USB storage device

This completes the configuration of the OTP. Shut down your Raspberry Pi and remove the SD card, which is no longer needed. Take the USB device you prepared in step 4 and insert it into one of the USB ports. Add the power supply and after a few seconds your Raspberry Pi will begin booting up. If you have a display connected you'll see the familiar rainbow splash screen appear. Note that the boot-up time may be slower than using an SD card – this depends on the type and speed of the USB drive you're using. However, once the Pi has completed booting up it will run at the usual speed.

10 Reusing the SD card

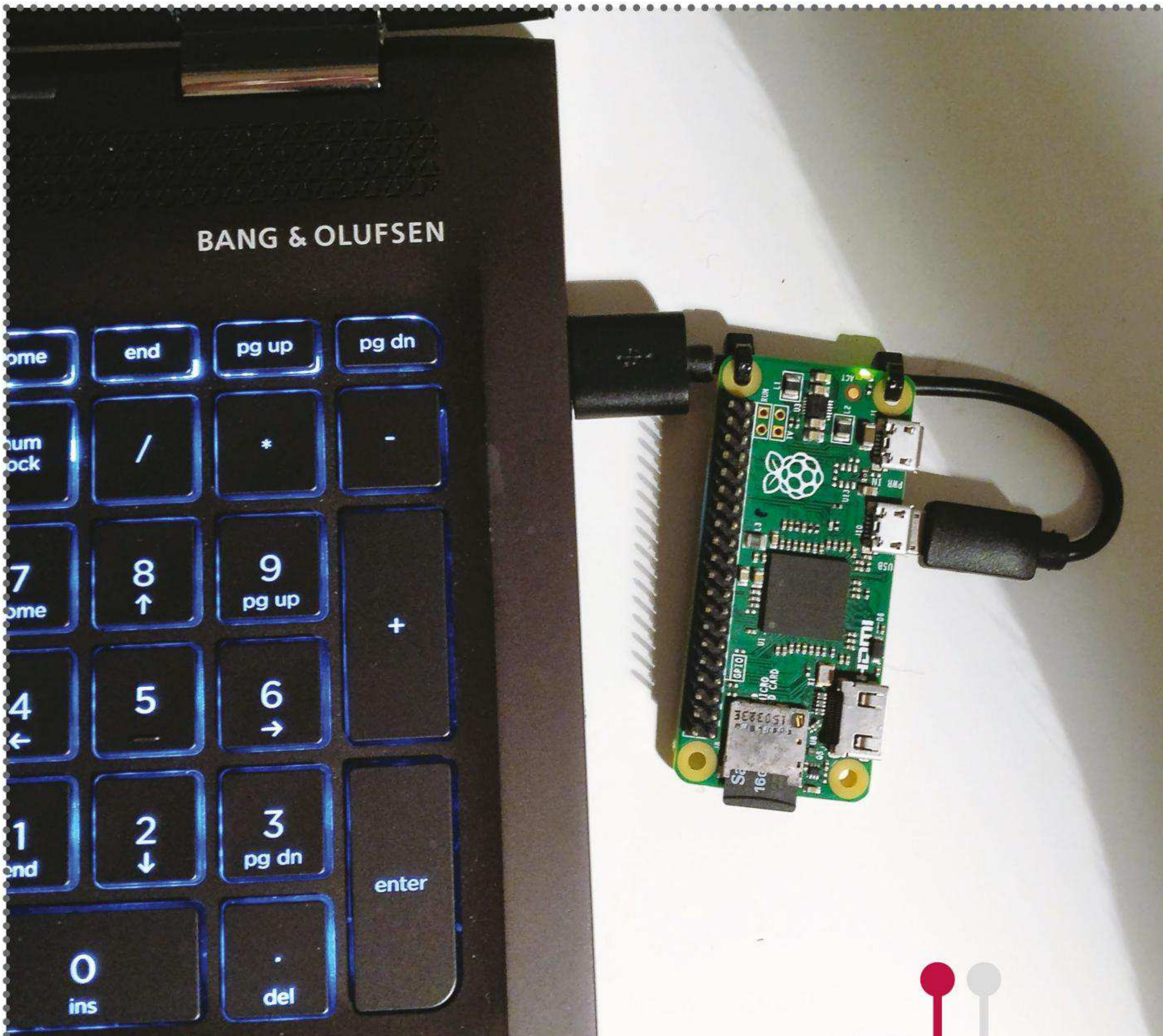
At some point in the future you will probably want to reuse the SD card that was used to set up the USB device. To do this you simply need to remove the line program_usb_boot_mode=1 line from config.txt so that the Raspberry Pi boots from the SD card. Since you can't now use this SD card to boot up the Pi you've just altered, you'll first need to insert the card into your main PC – don't use it yet with a different Pi if you have one, as when that one boots up, it too will be set to start from USB! Open the Terminal window and then open the config.txt file: sudo nano /boot/config.txt. Scroll down and locate the line of text program_usb_boot_mode=1 and either delete it or comment it out. You can now use this SD card in another Raspberry Pi.

The PoE HAT

The official Raspberry Pi PoE (Power over Ethernet) HAT is a new add-on board designed for the 3 B+ model that enables the Raspberry Pi to be powered via a power-enabled Ethernet network. It features 802.3af PoE, a fully isolated switched-mode power supply, 37–57V DC, a 25mm x 25mm brushless fan for processor cooling and fan control. Could this signal a move towards the Raspberry Pi being embedded in more IoT devices? You can purchase the PoE HAT from https://www.raspberrypi.org/products/poe-hat.

Access a Raspberry Pi Zero using a laptop

Configure OS settings and use the USB port to access both the command line and GUI from another computer



There's no doubt that the Raspberry Pi boasts a wide range of resources – software and hardware which can be used for computing, programming and creating exciting and engaging projects. There are numerous add-on boards and components to expand the capabilities of the Pi. A lot of these require access to the command line or the GUI via a screen, available in a range of sizes, styles and colours.

To use your Raspberry Pi you also require a keyboard, mouse, power supply or USB battery. The Raspberry Pi Zero also requires additional conversion sockets to add the various components. This often means you must carry around an additional kit if you want to access your Pi away from your desk or on the go.

This tutorial covers a step-by-step solution for using a USB cable and a few setup changes in order to configure your Raspberry Pi Zero to be accessible via the USB port of your laptop or device. Simply plug in your Pi, wait for it to boot up and then access it via the command line or the GUI: no need for an extra screen, keyboard, mouse or power supply. All code, projects and changes are saved directly to your SD card.

This makes it ideal for accessing them when travelling on a plane or train, or when you want to demonstrate a feature but don't have all the additional peripherals.

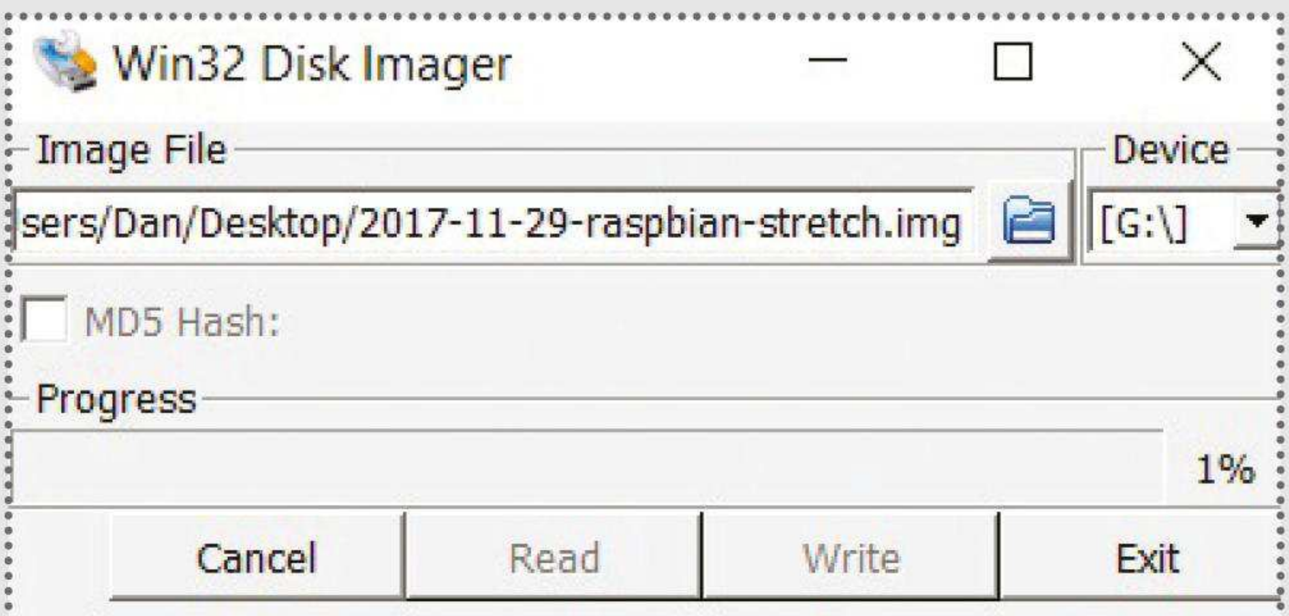
01 Getting started

Before we configure the settings to enable you to use and access your Raspberry Pi via the USB port, there are a few pieces of software to install. If you already have these, skip to step four. Depending on which operating

system you're using to access your Pi Zero, you might need to install the following additional software. For a computer running Windows, you'll need Bonjour, which is part of an iTunes install **(www.itunes.com)**. For a Mac OS or Linux PC, ensure the Avahi Daemon is installed. If you're using Ubuntu this is already built in.

02 Install Putty

To access the Pi Zero you also require a SSH client. You may already have one installed, or your OS may have one built-in. Putty is a popular free SSH client for Windows and can be downloaded and installed from www.ssh.com/ssh/putty/download. If you're using Linux, open the LX Terminal window and type sudo



aptitude install putty and sudo aptitude install putty-tools.

03 Install the Raspberry Pi OS

Begin this project with a fresh install of the current Raspberry Pi operating system, available from **www.raspberrypi.org/downloads/raspbian.** The file is compressed and it will need to be unzipped to extract the main .iso file. Then write this file to a blank microSD card using your normal method, or download Etcher

Access the GUI via USB

It's possible to access the desktop of your Pi using VNC. First load your Pi and select Menu > Preferences > Raspberry Pi Configuration. Click 'Interfaces' and set VNC to 'Enabled'. Now, on your laptop or computer, download and install the relevant Viewer from RealVNC: www.realvnc.com/en/connect/download/viewer. Plug the Pi into the laptop and wait for it to boot. Then using the VNC app enter the name raspberrypi.local and press Return. (You may be presented with an 'Identify check fail'; click Continue to log in.)

https://etcher.io – a simple and easy to use app for this.

04 Accessing the SD card

Once the OS has been written to the SD card, you're ready to start configuration. We recommended that you use **Notepad++** or a similar text editor rather than Windows' WordPad, as it lists the entries in the config file properly rather than as one continuous line of text. **Notepad++** can be downloaded from **https://notepad-plus-plus.org**. Once downloaded, open File Explorer and navigate to the SD card folder. You will see the two text files towards the top of the folder.

05 Enable SSH

Secure Shell (SSH) is a secure method of remotely logging into a network. The Raspberry Pi OS by default used to come with SSH enabled; however, this proved to be a security risk as many users didn't change the default credentials and therefore left their Pi open to

Access the GUI cont

Set the Encryption to 'Let VNC Server choose' and click Connect. Enter the user name pi and the password raspberry unless you've previously changed these in your setup. The desktop GUI will load up. You can adjust the window size and resolution in the VNC configuration settings.

```
# no display
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

dtoverlay=dwc2
```

unauthorised access. To enable the SSH, right-click and create a new file named ssh. Ensure that the file does not have any extensions (that is, no .txt). Save the file.

06 Edit the config.txt file

The first major change requires you to stipulate that you're using the dwc2 USB driver since this is where the Pi will be plugged in. Place the SD card into your computer and open it in File Explorer or equivalent program. Locate the file config.txt and open it. Scroll to the bottom of the file and on the next line down add the following line of code: **dtoverlay=dwc2**. Then save and close the file.

07 Edit the cmdline.txt file

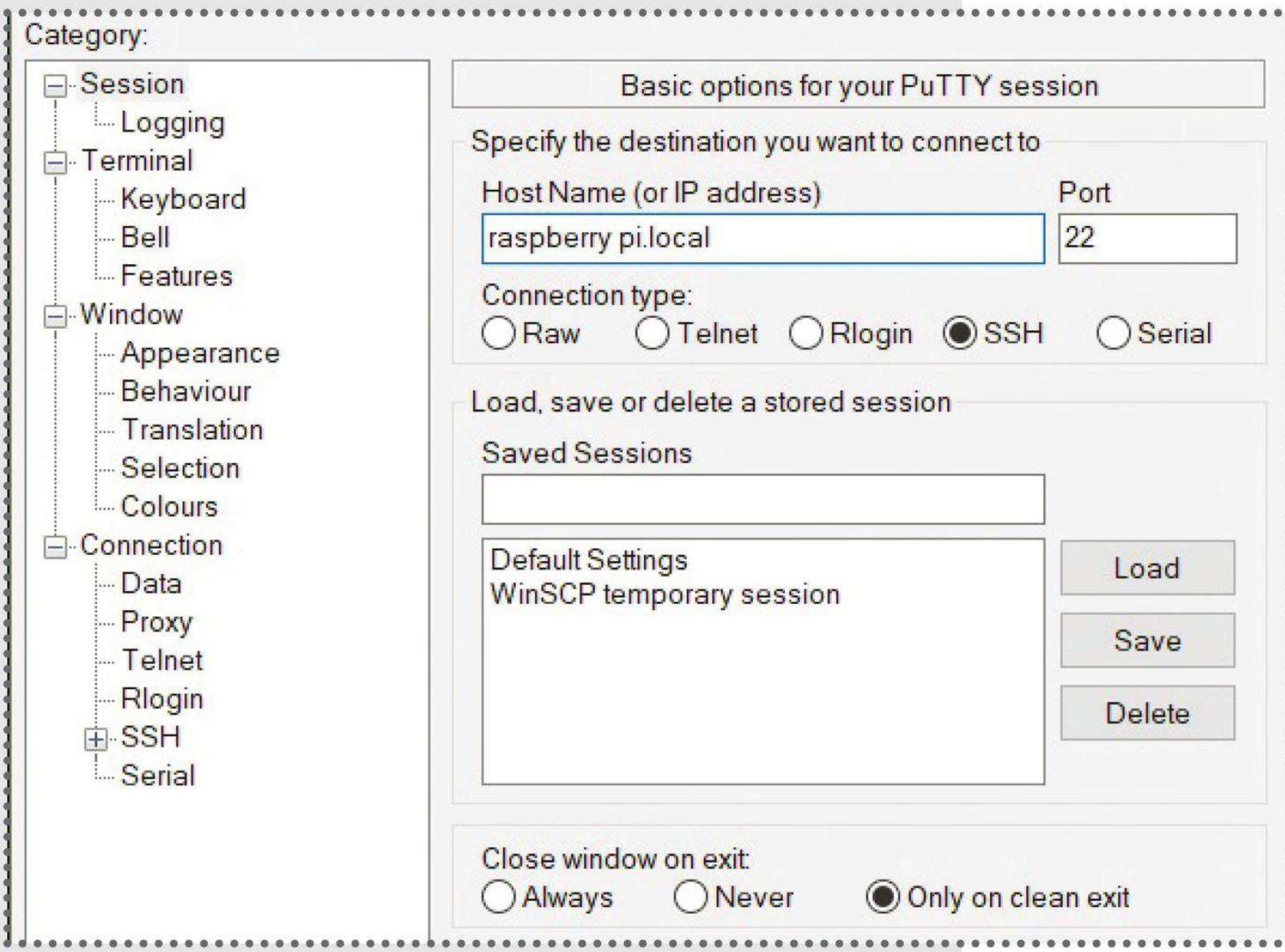
Locate and open the cmdline.txt file. This file contains a single line that includes several parameters, each separated by a single space. As there are no newlines in the file you must be precise when editing this code. Locate the word rootwait, then add a single space and enter the code **modules-load=dwc2,g_ether**. Add a single space after the entry and save and close the file.

08 Wire up the Pi

Once you've completed these changes you're ready to boot up your Raspberry Pi via the USB port. Eject the microSD card and put it into your Pi Zero. Take the Micro-USB to USB wire and attach it to the Micro-USB port on the Pi. Depending on the laptop's OS, you may need to turn off all networking connections; the easiest way to do this is to put the laptop into Flight mode. Now plug in the other end of the USB cable to one of the USB ports on your laptop. After about 90 seconds or so, the Raspberry Pi will be ready.

09 Accessing your Raspberry Pi

Once the Raspberry Pi has booted up, you can access it via Putty or another telnet program. Open Putty on your laptop and locate the 'Host Name' box at the top of the window. Enter the hostname raspberrypi.local, with the port number as the default 22. Select 'SSH' as the connection type. Press the Open button at the bottom of the window. You will be prompted to enter the username and password of your Raspberry Pi, which, unless you have changed it, will be pi and raspberry. Press Return, and you will be presented with the command line of your Raspberry Pi. Obviously you only need to make these changes once, and from now on you can use your laptop to access the Pi.

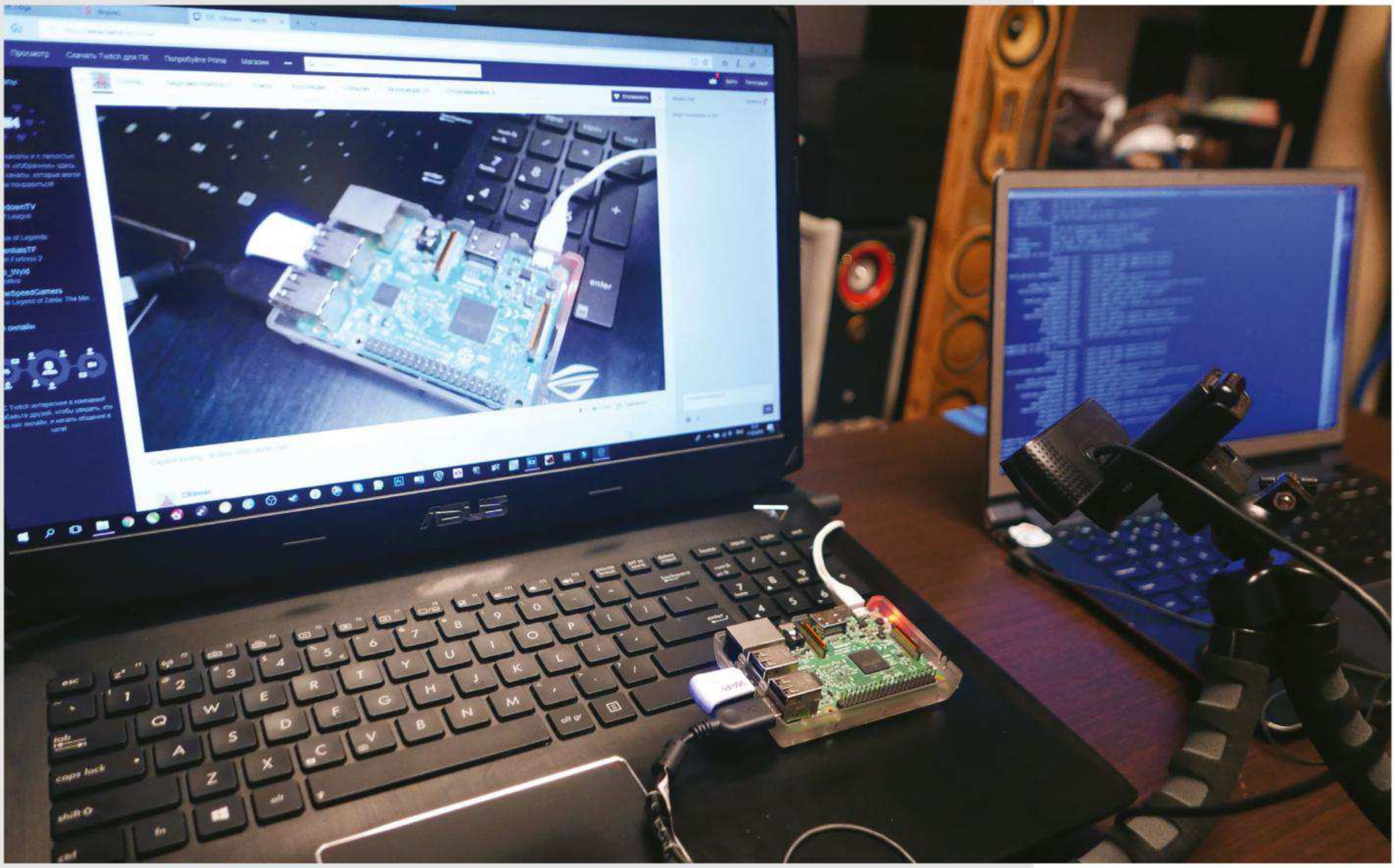


Stream to Twitch with a Pi

Enable non-conventional uses for streaming with one of the most popular single-board computers

On a hackathon a year ago, we decided to build a 24/7 life-streaming device. Among other things, we learned how to create a Twitch-streaming device from a Raspberry Pi – and now so can you! With cheap high-res camera sensors, fast and affordable internet connections and hardware-accelerated HD video encoding, it's no wonder live video-streaming blew up. People stream games, real-life events, tutorials and even the most boring things, like driving to grab some food from McDonald's – there are new purposes for streaming being invented every day, and if you find a novel and interesting subject, viewers will come.

Talking of viewers, they are one important aspect of modern livestreaming: find your audience and interact with it. On streaming platforms, there's usually a chat that viewers can use to message you in real time, as opposed to traditional one-way live streams. Even if interacting with people isn't your goal, the chat still has a lot of potential for different ideas and use cases, from remote-controlled robots to installing



operating systems.

You, too, can grab a webcam and stream. First and foremost, you can share your experiences with others – be it a video game you're playing, an event you're attending or something you just want to talk about. However, you don't have to get involved in the stream personally. How about setting up a kitchen camera at work, so that you can see when the coffee pot is full and you can go grab a coffee? Incidentally, this is how internet webcams first became popular, back in 1991. How about remotely checking for a free space in the car park? Alternatively, why not put a webcam on your first robot, and maybe let the viewers take control of it? Granted, the last one sounds like a terrible idea, but experience shows it's more likely to be fun than disastrous.

Below You too can start streaming, using just a Raspberry Pi, a USB camera and an internet connection

With a streaming platform backing you, you can do all this and more. There are also technical reasons for using a platform. Platforms such as YouTube, Twitch and others enable you to stream your content in a way that you yourself cannot – particularly when it comes to assisting you in building a large audience of loyal viewers. For that, they need you to send your stream to their server, which then re-streams your video to everybody willing to watch.

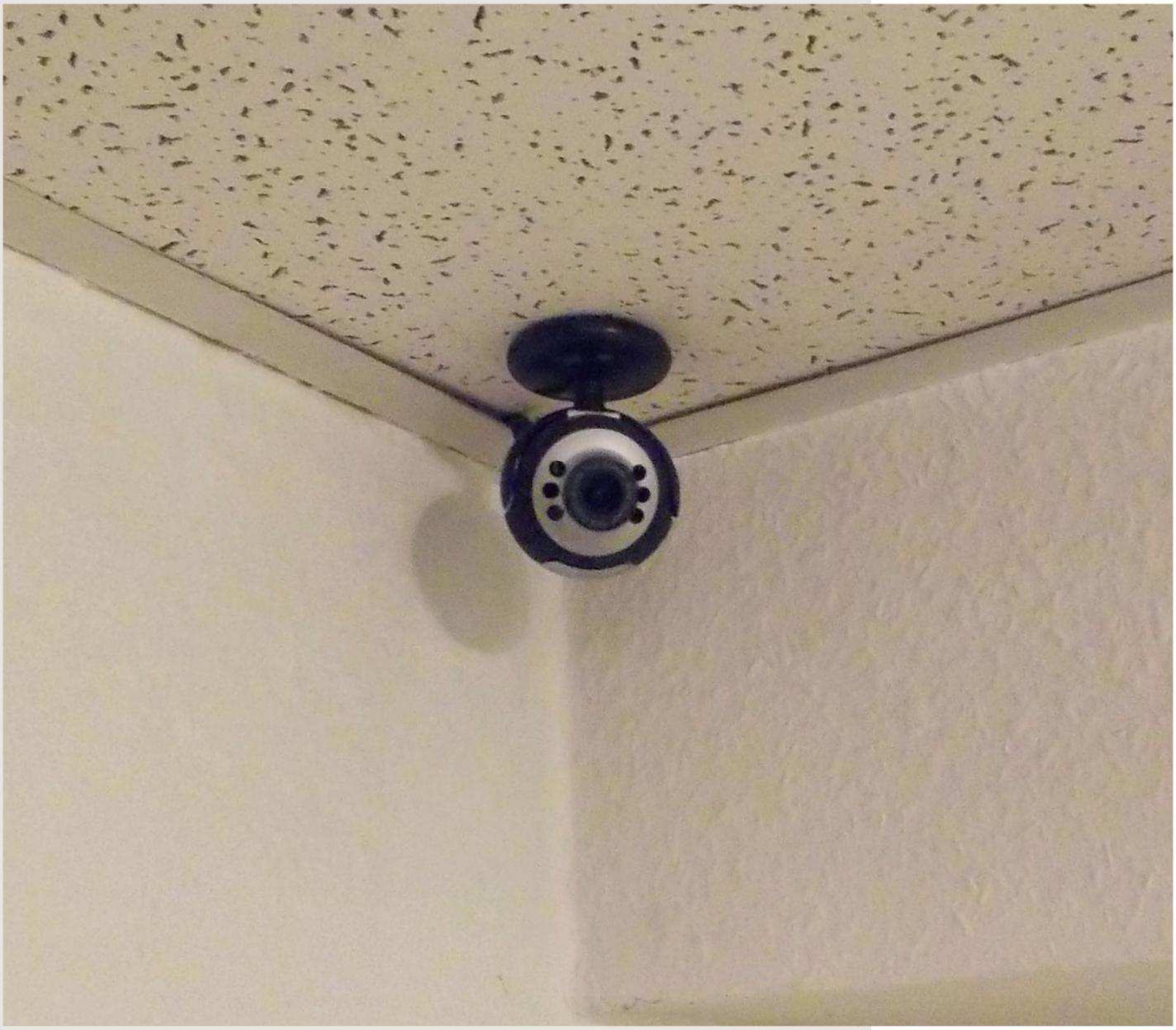
Compared to the approach where each viewer connects to your computer to fetch the stream, this reduces your internet usage dramatically (especially when your stream is viewed by hundreds or thousands of people). As a consequence, this improves latency; even though your stream has to travel through another server first, that server can handle all the viewers better than your home router can. Don't forget that streaming platforms also provide many other useful features such as webchats, subscriptions (even paid ones), landing pages for new viewers, game integration and so on.

You might not need a platform for your streaming, though. If you're streaming mostly static content such as a security camera, trying to use it as a video-call, or setting up a local stream such as the aforementioned 'coffee pot watcher', you might be better off using something else such as voice-call software or a stream that's limited to your local network. Don't dismiss this tutorial though, as it dives into technical details you'll likely find useful either way.

Thanks to open source software, you can build your own streaming setup cheaply and easily, with

Chatbot pizazz

You can run the chatbot on the same Pi from which you're streaming. With a little bit of additional hardware, you can enable your viewers to control camera parameters such as focus or exposure, adjust lighting, or even show their messages on some small display – letting them improve the stream themselves!



just a Raspberry Pi and a camera attached to it. With help from one of the popular video processing and transmission toolkits such as GStreamer or FFmpeg, you can stream video to Twitch, YouTube and Facebook, making only small changes to the script itself. So how are all these different platforms supported without the need for significant changes? They all use the same standard for streaming, called RTMP (Real-Time Messaging Protocol). Open source

Above A Pi-based security camera, streaming to an offsite server that stores recordings

software can take a video stream from your webcam and re-stream it to a server accepting RTMP.

RTMP only covers communications, not video and audio formats. As the server needs to send the same format to all its clients, it's also picky about the way the incoming video is encoded; the more video encodings that a server is able to receive, the more complicated (and slower) the server software needs to be. Twitch servers define a format that you need to use, so you need to convert your video to this format before sending it to the server, and that will be a big part of what we're doing.

"Thanks to open source software, you can build your own streaming setup cheaply and easily"

Check the requirements

The scripts will be tailored for a Raspberry Pi; to be specific, we will be using Raspberry Pi hardware-accelerated encoding capabilities. However, we'll explain everything along the way so that you can tweak the code to your own needs, including running it on another board that doesn't have hardware acceleration. You will want a Raspberry Pi 2 or 3; lower-end models such as the Pi B, B+ and Zeros can generally handle streaming, but the experience won't be as smooth, especially if sound is involved. Also, don't forget to increase your GPU RAM to at least 128MB using raspi-config.

We will also be using Raspbian, the Raspberry Pi flavour of Debian. You might be using some other kind of Linux distribution, but it's likely that the instructions will apply to you, too; however, you might find that some Raspberry Pi-specific plug-ins might be missing. In that case, let your favourite search engine help you out.

Picking the right kind of webcam is very important. Webcams aren't simple; inside, they have specialised chips that not only grab and transfer the image, but also filter, adjust and encode it, which is important when you're trying to stream a detailed image of what's happening. Doing those things in hardware offloads your Pi's CPU. This is one of the most significant differences between cheap and expensive cameras. Personally, we have had a great experience with Logitech cameras, but you might find another brand that's as suitable or even better. You can also use the official Raspberry Pi camera together with the bcm2835-v4l2 driver.

The resolution of the stream has to be one that your webcam supports, and different resolutions can influence your stream's quality a lot. If you want to see which resolutions your webcam is capable of, run v4l2-ctl --list-formats-ext. You might need to install it beforehand: sudo apt install v4l-utils.

So fundamentally, what do we need to stream video from a webcam? First, capture the video stream in the format that our camera exposes, then convert it to the format that the streaming servers use, then send the resultant video to a streaming server. There are multiple Linux toolkits that can be used for this, the most popular being FFmpeg and GStreamer. If you're using a user-friendly streaming application, it's likely that it has one of these tools under the bonnet. Let's focus on FFmpeg, and try streaming to Twitch from a USB webcam, add sound to our stream, and then create our own Twitch bot.

FFmpeg inputs

FFmpeg supports many kinds of inputs. You can stream from a video file, share your desktop with your viewers or rebroadcast a local network source, for example. It can also do various transformations to the video, in case you put the webcam in your robot upside-down or need to crop the resulting image.

FFmpeg is a media processing toolkit. It can be used for audio and video capture, conversion, extraction and repackaging, as well as streaming, so its covers everything we need. FFmpeg isn't available for download on Raspbian Jessie as it was replaced with avconv, but is available on Raspbian Stretch. If your /etc/os-release file shows that you still have Raspbian Jessie (VERSION_ID="8") installed, we suggest you upgrade your system (or compile FFmpeg yourself if you can't).

If you're on Raspbian Stretch, installing FFmpeg should be as simple as sudo apt install FFmpeg. To check that you have FFmpeg installed, run FFmpeg -version. It should start with "FFmpeg version 3.2.10" – make sure it's not significantly older.

Above Pi Zero W and USB webcam form a $20 wearable setup for streaming to Twitch

Streaming: video only

First, let's cover a simple streaming case: no sound, just video. We have a USB camera (available as /dev/video0), and we have a Twitch RTMP URL, to which we should send our stream. A basic FFmpeg command line that does the job is as follows:

```
FFmpeg -hide_banner -f v4l2 -s 1280x720 -r 4 -i /dev/video0 -vcodec h264_omx -g 8 -keyint_min 4 -b:v 500k -minrate 100k -maxrate 500k -pix_fmt yuv420p -bufsize 500k -preset veryfast -f flv "rtmp://live.twitch.tv/app/live_864954762_J2e7818OEnOIve2qdUIWAKwuNlQ4weo"
```

That's a lot of parameters for a single command! Let's go through it so that you understand what's going on. Here's the template for our command line:

```
FFmpeg {global options} -f {input type} {input options} -i {input} {codec} {codec options} -f {output type} {output destination}
```

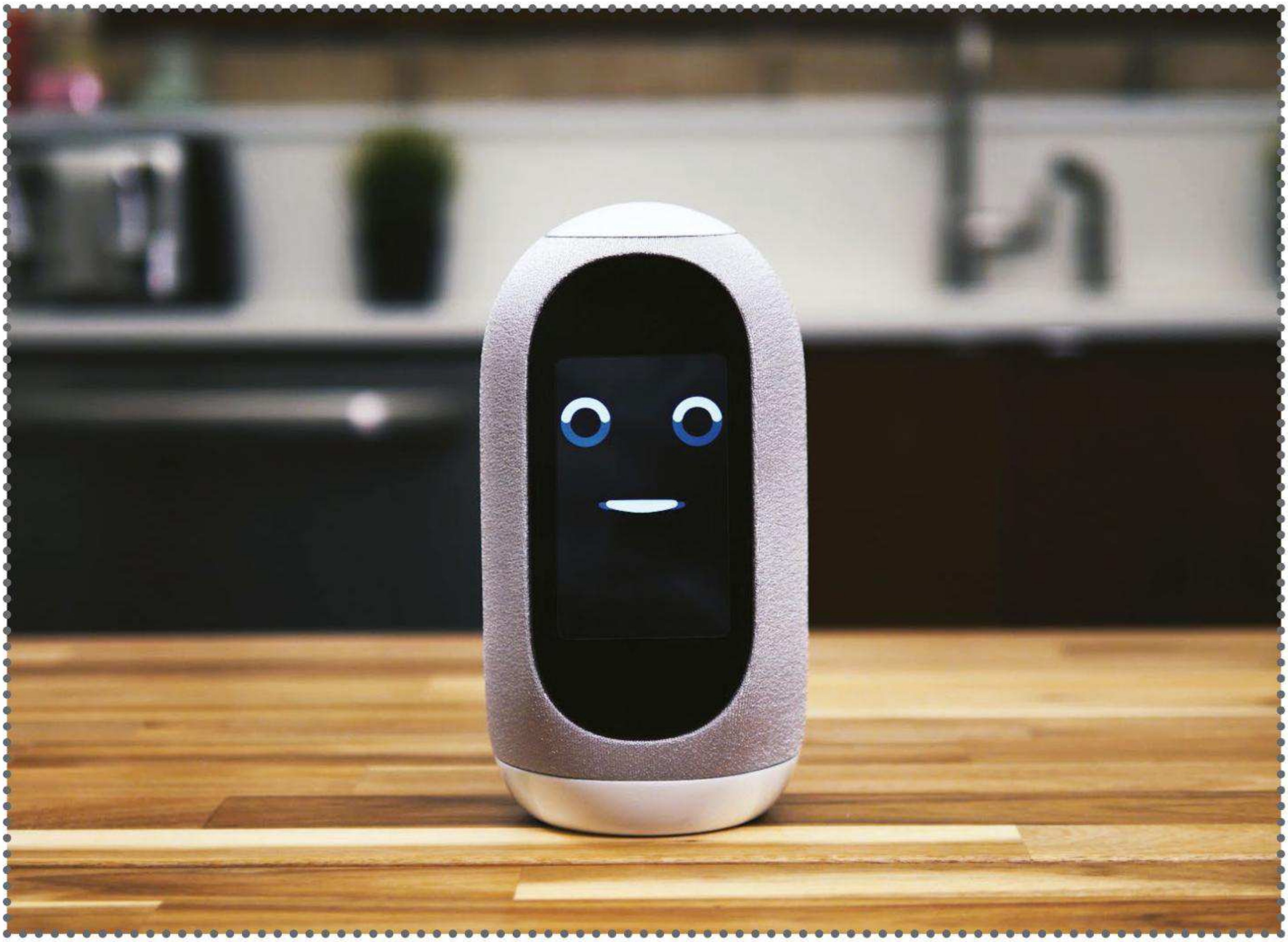
We're only using one global option: -hide-banner, which tells FFmpeg not to print its version information on start. Our webcam is /dev/video0; in your case, it might end with another number, but you will notice if it's a wrong one. To capture the video itself, we're using the Video4Linux system and its FFmpeg plugin called v4l2, telling it the resolution to use with -s and FPS (frames per second) that we need with -r. In our case, it's 4fps.

Twitch requires that we compress our video as

Next issue

Get inspired · Expert advice · Easy-to-follow guides

"Create your own voice assistant"



Get this issue's source code at:

www.linuxuser.co.uk/raspicode